

The Diagnostic Manager

(diag_manager)

Author: Seth Underwood (DRC/HPTg)
Phone: +1 - 609 - 987 - 5058
Fax: +1 - 609 - 987 - 5063
Email: Seth.Underwood@noaa.gov

National Oceanic and Atmospheric Administration
Geophysical Fluid Dynamics Laboratory
Princeton, NJ 08542
<http://www.gfdl.noaa.gov>



Outline

- Overview
- Diagnostic Manager Routines
- Diagnostic Manager Namelist Options
- Diagnostic Table
 - diag_table_chk
 - Exercise



Outline

- Overview
- Diagnostic Manager Routines
- Diagnostic Manager Namelist Options
- Diagnostic Table
 - diag_table_chk
 - Exercise



Overview

- A set of simple calls for parallel diagnostics on distributed systems
- Provides a convenient set of interfaces for writing data to disk
- Geared toward writing data in NetCDF format
- Build on the parallel I/O interface of FMS code (shared/mpp/mpp_io.F90)
- Provide data to disk at any number of sampling and/or averaging intervals specified at run-time
- Run-time specification of diagnostics controlled through the diag_table file



Features

- Output scalar (0D) to 3D fields
- Calculate time average,
minimum/maximum values
- Output data globally or regionally



Outline

- Overview
- Diagnostic Manager Routines
- Diagnostic Manager Namelist Options
- Diagnostic Table
 - diag_table_chk
 - Exercise



Usage

1. Call `diag_manager_init` to initialize the diagnostic manager
2. Register output field axes with `diag_axis_init`
3. Call `register_diag_field` on all output fields
4. Call `send_data` to “send data” to the diagnostic manager
5. Call `diag_manager_end` prior to ending the model to flush all buffered writes to disk



Usage

1. Call `diag_manager_init` to initialize the diagnostic manager
2. Register output field axes with `diag_axis_init`
3. Call `register_diag_field` on all output fields
4. Call `send_data` to “send data” to the diagnostic manager
5. Call `diag_manager_end` prior to ending the model to flush all buffered writes to disk



diag_manager_init

```
SUBROUTINE diag_manager_init (diag_model_subset, err_msg)
```

- Called during initialization of Model
 - Can be called multiple time safely
 - Only first call is executed
- Read in diag_table, setup files and input fields



diag_manager_init

```
SUBROUTINE diag_manager_init (diag_model_subset, err_msg)
```

From **coupler/coupler_main.F90**

```
use diag_manager_mod, ONLY : diag_manager_init  
  
...  
  
if ( Atm%pe )then  
    ! change diag_model_subset from DIAG_ALL  
    call mpp_set_current_pelist (Atm%pelist)  
    if( atmos_npes /= npes ) diag_model_subset = DIAG_OTHER  
else if ( Ocean%is_ocean_pe ) then  
    ! change diag_model_subset from DIAG_ALL  
    call mpp_set_current_pelist (Ocean%pelist)  
    if ( ocean_npes /= npes ) diag_model_subset = DIAG_OCEAN  
end if  
call diag_manager_init (DIAG_MODEL_SUBSET=diag_model_subset)
```



diag_axis_init

```
INTEGER FUNCTION diag_axis_init (name, data, units, cart_name,&  
    & long_name, direction, set_name, edges, Domain,&  
    & Domain2, aux, tile_count)
```

- Initialize the axis, and return the axis ID
- Use the returned ID in calls to
`register_diag_field`



diag_axis_init

```
INTEGER FUNCTION diag_axis_init (name, data, units, cart_name,&
    & long_name, direction, set_name, edges, Domain,&
    & Domain2, aux, tile_count)
```

- From

[atmos_cubed_sphere/tools/fv_dynamics.F90](#)

```
use diag_manager_mod, ONLY : diag_axis_init
...
do n = 1, ntileMe
    field = 'grid'
    id_x = diag_axis_init ('grid_x', grid_x, 'degrees_E',&
        & 'x', 'cell corner longitude', set_name=trim(field),&
        & Domain2=Domain, tile_count=n)
    id_y = diag_axis_init ('grid_y', grid_y, 'degrees_N',&
        & 'y', 'cell corner latitude', set_name=trim(field),&
        & Domain2=Domain, tile_count=n)
end do
id_phalf = diag_axis_init('phalf', phalf, 'mb', 'z',&
    & 'ref half pressure level', direction=-1,&
    & set_name="dynamics")
axes(1) = id_xt
axes(2) = id_yt
axes(3) = id_pfull
```



register_diag_field

```
INTEGER FUNCTION register_diag_field (module_name, field_name, [axes,] &  
    & init_time, long_name, units, missing_value, range, [mask_variant], &  
    & standard_name, verbose)
```

- Scans input fields from diag_table
- Setup output field buffers for all input fields
- Use returned ID in send_data calls



register_diag_field

```
INTEGER FUNCTION register_diag_field (module_name, field_name, [axes,&
& init_time, long_name, units, missing_value, range, [mask_variant],&
& standard_name, verbose)
```

From atmos_param/radiation_driver/radiation_driver.F90:

```
use diag_manager_mod, ONLY : register_diag_field  
  
...  
  
id_pt = register_diag_field ('dynamics', 'temp', &  
& axes(1:3), Time, 'temperature', 'K', &  
& missing_value=missing_value, range=trange )
```



send_data

```
LOGICAL FUNCTION send_data (diag_field_id, field, time,&  
  & is_in, js_in, ks_in, mask, rmask, ie_in, je_in,&  
  & ke_in, weight, err_msg)
```

- Pass data to `diag_manager_mod` for output to file
- Simple time operations done if indicated in `diag_table`



send_data

```
LOGICAL FUNCTION send_data (diag_field_id, field, time, &
    & is_in, js_in, ks_in, mask, rmask, ie_in, je_in, &
    & ke_in, weight, err_msg)
```

- From `atmos_cubed_sphere/tools/fv_diagnostics.F90`

```
use diag_manager_mod, ONLY : send_data

...
do n = 1, ntileMe
    if ( id_pt > 0 ) used = send_data(id_pt, Atm(n)%pt, &
        & (isc:iec,jsc:jec,:), Time)
end do
```



diag_manager_end

SUBROUTINE `diag_manager_end` (`time`)

- Flush diagnostic buffers
- Close diagnostic files
- A warning for each field in the `diag_table` that has not been registered in `register_diag_field`



diag_manager_end

SUBROUTINE **diag_manager_end** (**time**)

- From `coupler/coupler_main.F90`

```
use diag_manager_mod, ONLY : diag_manager_end  
  
...  
  
call diag_manager_end (Time)
```



Outline

- Overview
- Diagnostic Manager Routines
- Diagnostic Manager Namelist Options
- Diagnostic Table
 - diag_table_chk
 - Exercise



diag_manager_nml

- Namelist options to control behavior of diag_manager
- Namelist options defined/modified in XML tag
`<namelist name="diag_manager_nml">`

```
<experiment name="experiment_name">  
    ...  
    <input>  
        <namelist name="diag_manager_nml">  
            mix_snapshot_average_field = .TRUE.  
            max_input_fields = 400  
            max_output_fields = 500  
        </namelist>  
    </input>  
</experiment>
```



diag_manager_nml

- **INTEGER :: max_files**
 - Maximum number of files allowed in diag_table
 - Default :: 31
- **INTEGER :: max_output_fields**
 - Maximum number of output fields allowed in diag_table
 - Default :: 300
- **INTEGER :: max_input_fields**
 - Maximum number of registered fields allowed
 - Default :: 300
- **LOGICAL :: do_diag_field_log**
 - Write out all registered fields to a log file
 - Default :: .FALSE.



diag_manager_nml

- **LOGICAL :: use_cmor**
 - Override the `missing_value` to the CMOR value of `-1.0e20`
 - Default :: .FALSE.
- **LOGICAL :: issue_oor_warnings**
 - Issue a warning if a value passed to `diag_manager` is outside the given range
 - Default :: .TRUE.
- **LOGICAL :: oor_warnings_fatal**
 - Treat out-of-range errors as FATAL
 - Default :: .FALSE.



Outline

- Overview
- Diagnostic Manager Routines
- Diagnostic Manager Namelist Options
- Diagnostic Table
 - diag_table_chk
 - Exercise



diag_table

- All data written out by diag_manager is controlled via the diag_table
- The diag_table is a plain text file that lists files and fields, and the frequency of output
- The diag_table contains the "base date" used by the model
- Note: None of the lines in the diag_table can span multiple lines
 - Lines are split here for readability ONLY



diag_table

```
AM3_Class
1 1 1 0 0 0

# Output files
"atmos_daily", 24, "hours", 1, "days", "time"

# Output fields
"dynamics", "ps",      "ps",      "atmos_daily", "all", .true.,   "none",  2
"dynamics", "sphum",    "sphum",    "atmos_daily", "all", .true.,   "none",  2
"dynamics", "temp",     "temp",     "atmos_daily", "all", .true.,   "none",  4
"dynamics", "ucomp",    "ucomp",    "atmos_daily", "all", .true.,   "none",  4
"dynamics", "vcomp",    "vcomp",    "atmos_daily", "all", .true.,   "none",  4
```



diag_table -- Header

```
AM3_Class
```

```
1 1 1 0 0 0
```

- First line must be a string describing the experiment
 - Can be any arbitrary string (no spaces allowed)
 - AM3_Class (Good)
 - AM3 Class (Bad)
- Second line must be the base date of the experiment
 - The base date is the reference time used for the time units, and must be the same or earlier than the model start time.
 - 6 integers separated by spaces
 - Note: A value of zero for Year, month and day is not valid.
 - year month day hour minute second
 - 1 1 1 0 0 0



diag_table -- File Description

```
# Output files  
"atmos_daily", 24, "hours", 1, "days", "time"
```

- File descriptions contain 6 required
- Can be intermixed with field descriptions, but a file must be defined before it can be used by a field



diag_table -- File Description

```
# Output files  
"atmos_daily", 24, "hours", 1, "days", "time"
```

- **CHARACTER(len=128) :: file_name**
 - Output name without the trailing ".nc".
 - A single file can produce multiple files using special time string keywords (%#tt)
 - # is a mandatory single digit number specifying the width of the field
 - tt is either 'yr', 'mo', 'dy', 'hr', 'mi', 'sc'.
 - "file2_yr_dy%1yr%3dy" yields "file2_yr_dy_1_001" (for the file created on year 1 day 1 of the model run)



diag_table -- File Description

```
# Output files  
"atmos_daily", 24, "hours", 1, "days", "time"
```

- **INTEGER :: output_freq**
 - How often to write fields to the file
 - Can be '-1', '0', or '> 0'
 - -1 :: Output at the end of the run only
 - output_freq_units is ignored
 - 0 :: Output every time step.
 - output_freq_units is ignored
 - >0 :: Output frequency in output_freq_units
- **CHARACTER(len=10) :: output_freq_units**
 - Time unit for output
 - Can be "years", "months", "days", "minutes", "hours" or "seconds"



diag_table -- File Description

```
# Output files  
"atmos_daily", 24, "hours", 1, "days", "time"
```

- **INTEGER :: file_format**
 - Output file format.
 - Only valid option is '1' for NetCDF file output
- **CHARACTER(len=10) :: time_axis_units**
 - Time units for the output file time axis.
 - Can be "years", "months", "days", "minutes", "hours" or "seconds"
- **CHARACTER(len=128) :: time_axis_name**
 - Time axis name
 - Is usually just "time"



diag_table -- File Description

```
# Output files  
"atmos_other", 5, "days", 1, "months", "time"
```

- Filename?
 - atmos_other
- Output frequency?
 - once every 5 days
- Time axis name?
 - time
- Time axis units?
 - months



diag_table -- Field Description

```
# Output fields  
"dynamics", "temp",      "temp",      "atmos_daily", "all", .true., "none", 2
```

- Field lines contain 8 fields
- Can be intermixed with file descriptions, but the file must be defined prior to the use in a field
- Field line can contain fields not written to any file
 - File name must be null



diag_table -- Field Description

```
# Output fields  
"dynamics", "temp",      "temp",      "atmos_daily", "all", .true., "none", 2
```

- **CHARACTER(len=128) :: module_name**
 - Module that contains the field_name variable
 - Module name defined in register_diag_field call
- **CHARACTER(len=128) :: field_name**
 - Module variable name that has data to be written to file
 - field_name defined in register_diag_field call
- **CHARACTER(len=128) :: output_name**
 - Name of the file as written in file_name
 - Useful if needing field name to match a different output standard



diag_table -- Field Description

```
# Output fields  
"dynamics", "temp",      "temp",      "atmos_daily", "all", .true., "none", 2
```

- **CHARACTER(len=128) :: file_name**
 - Name of the file where the field is to be written
 - Must match file_name in file description
- **CHARACTER(len=50) :: time_sampling**
 - Currently not used. Please use string "all"



diag_table -- Field Description

```
# Output fields  
"dynamics", "temp",      "temp",      "atmos_daily", "all", .true., "none", 2
```

- **CHARACTER(len=50) :: reduction_method**
 - Data reduction method to perform prior to writing data to disk
 - .TRUE., "average"
 - Average over length of file writes (defined in file description)
 - .FALSE., "none"
 - No reduction performed. Write current time step value
 - "min", "max"
 - Write minimum / maximum value over length of file writes



diag_table -- Field Description

```
# Output fields  
"dynamics", "temp",      "temp",      "atmos_daily", "all", .true., "none", 2
```

- **CHARACTER(len=50) :: regional_section**
 - Bounds of the regional section to capture
 - Each region must be written to a separate file
 - "none" indicates a global region
 - "lon_min, lon_max, lat_min, lat_max, \
 vert_min, vert_max"
 - Use **vert_min=vert_max=-1** for all vertical levels
 - Use **lon_min=lon_max=lat_min=lat_max=-1** for vertical sub-region
 - Use **lon_min=lon_max** and **lat_min=lat_max** for single column



diag_table -- Field Description

```
# Output fields  
"dynamics", "temp",      "temp",      "atmos_daily", "all", .true., "none", 2
```

- "**lon_min, lon_max, lat_min, lat_max, \ vert_min, vert_max**"
 - "25.5, 94.3, -60.0, -55.0, 100, 920"
- Use **vert_min=vert_max=-1** for all vertical levels
 - "50.0, 52.0, 45.5, 46.5, -1, -1"
- Use **lon_min=lon_max=lat_min=lat_max=-1** for vertical sub-region
 - "-1, -1, -1, -1, 150, 850"
- Use **lon_min=lon_max** and **lat_min=lat_max** for single column
 - "25.5, 25.5, 66.5, 66.5, -1, -1"



diag_table -- Field Description

```
# Output fields  
"dynamics", "temp",      "temp",      "atmos_daily", "all", .true., "none", 2
```

- **INTEGER :: packing**
 - Fortran number KIND of the data written
 - Valid values
 - 1 :: double precision
 - 2 :: float



diag_table -- Field Description

```
# Output fields
"dynamics", "ps", "ps_a", "atmos_other", "all", .false.,
  "15.0, 20.0, -5.0, 5.0, -1, -1", 2 \
```

- Filename?
 - atmos_other
- Variable is from module?
 - dynamics
- Variable name?
 - ps
- Name of variable in output file?
 - ps_a
- Regional or global?
 - Regional
- What is the region?
 - 15.0-20.0 Longitude, -5.0-5.0 Latitude, all vertical levels
- Reduction routine?
 - None, instantaneous output written every time file atmos_other written



diag_table_chk

- Utility to check diag_table for errors
- Prints number of files, fields, etc
 - May be useful if using a large diag_table
- Run "diag_table_chk -h" for help
- Common errors
 - Missing comma (,)
 - Misspelled filename
 - File used in field before file defined
- Does not check if field names valid
 - diag_manager_mod warns of un-registered fields at end of run



Exercise

1. Copy /ncrc/home2/Seth.Underwood/ssam2012/diag_table to your home directory, and do the following.
2. Add new global field, with monthly average
 1. Module: moist
 2. Variable: precip
 3. Output Name: precip
3. Add new file with 8 daily outputs
4. Add new global field with 8 daily instantaneous output
 1. Module: dynamics
 2. Variable: tm
 3. Output Name: tm
5. Add new global field with a daily max value
 1. Module: flux
 2. Variable: wind
 3. Output Name: wind_max
6. Add a new file for regional output with output written once every 5 days
7. Add a new regional variable to the file you created in 5.
 1. Module: radiation
 2. Variable: olr
 3. Output Name: olr
 4. Longitude Range: 20E - 60E
 5. Latitude Range: -60N - -30N
8. Run your diag_table through diag_table_chk, and fix all error(s), and answer the following questions.
 1. How many files?
 2. How many variables?

